Pre-print submitted to Applied Numerical Mathematics

# Monotone Multivariate Interpolation of Scattered Data Using Nested Hypercubes

Scott M. Murman[*]

*NASA Ames Research Center, Moffett Field, CA, USA*

**Abstract**

One-dimensional linear interpolation is extended to arbitrary dimensions and scattered data using nested hypercubes. This is targeted at the evaluation of aerodynamic performance data in trajectory simulations and the generation of multi-fidelity response surfaces, though the approach is general. The algorithm demonstrates logarithmic scaling and quadratic convergence for regular and scattered interpolation centers. The performance for synthetic aerodynamic data, including steep gradients near the sonic line, is also provided. The efficiency of multi-threaded parallel implementations in a shared-memory arena, including the use of graphical processing units, is demonstrated.

## Nomenclature

| | |
|---|---|
| $\phi$ | radial basis function |
| $d$ | dimensions of the parameter space |
| $m$ | number of functionals |
| $n$ | number of interpolation centers |
| $p$ | number of partitions |

## 1 Introduction

The choice of an interpolation method for multivariate scattered data depends in large part upon the features of the data and the intended application[1]. The primary motivation for the current work is the evaluation of aerodynamic performance data in trajectory simulations, though the

[*]Scott.M.Murman@nasa.gov

approach is general and has broad applicability. The parameter space for this application has high dimensionality, typically $d \geq 6$, as a flight vehicle has a minimum of six degrees of freedom, and often more when controllers are considered. The data contains multiple uncorrelated functionals, again a minimum of six for the general case, and is naturally clustered near design points and steep gradients. It is common to encounter discontinuous data, for example near the sonic line or when using step controllers, so that a monotonic interpolation is required. Monte Carlo analysis or design optimization can generate millions of trajectory simulations which exercise obscure regions of the aerodynamic database, often requiring extrapolation (*cf.* [2] for an example). Thus we require an efficient interpolation method which scales well for both high dimensionality and number of interpolation centers ($n$), can handle highly clustered data, while still being monotonic and supporting extrapolation.

Current methods for handling this type of aerodynamic performance data are typically restricted to partitioned structured data[3]. This approach is difficult to automate, and cannot fully leverage Design of Experiments methodology, placing too much emphasis on off-design or benign conditions. This lack of flexibility imposes inefficiencies upstream into the most expensive part of the process, the data generation itself, either with physical testing (*e.g.* wind tunnels) or computational simulations.

Global methods for interpolation based on radial distance, such as radial basis functions (RBF) or Kriging, are popular for smooth, evenly-spaced data sets. These methods have the potential for high accuracy[4], but are expensive, in terms of storage (typically $\mathcal{O}(n^2)$) and complexity (typically $\mathcal{O}(n \log n)$). The weight calculation for radial distance methods can become numerically stiff for large $d$, and preconditioning tuned to the individual dataset is common to achieve consistent performance[5]. These are non-monotonic methods, which can produce excessive Gibbs phenomena for clustered or non-smooth data, and can be unsuitable for extrapolation. Further, a separate weight calculation is necessary for each functional.

Alternatively, local simplicial methods (tesselation) can provide a monotonic interpolation independent of the number of functionals, and are less sensitive to clustering. However, these methods become prohibitively costly in higher dimensions (approaching $\mathcal{O}(n^{\lfloor d/2 \rfloor - 1})$ complexity)[6, 7], and require special handling for extrapolation.

The current work describes a local method for efficiently constructing a convex hull surrounding an interpolation point which scales well to high dimensions and number of interpolation centers (optimal in storage and $\mathcal{O}(2^d \log n)$ in complexity), while still allowing extrapolation without special handling. Clustered data is naturally handled in the convex hull construction. The algorithm is outlined in the next section, followed by numerical examples to document the accuracy and efficiency for practical problems.

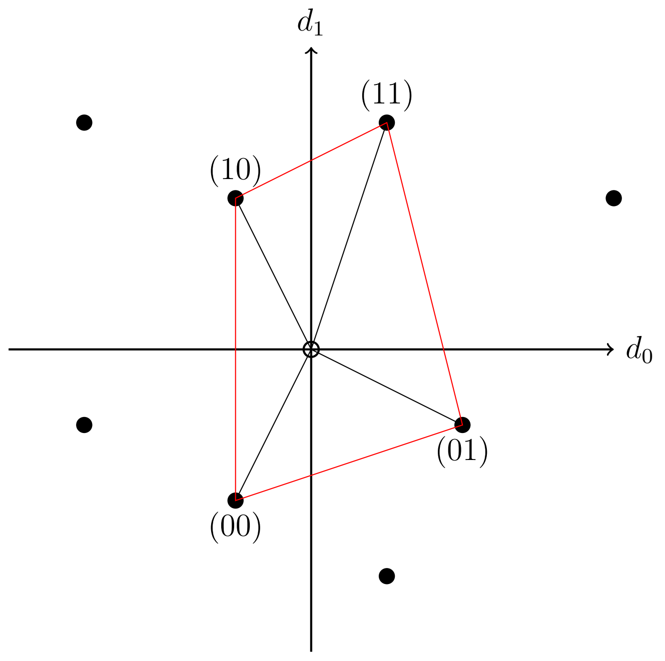


**Figure 1:** Bins of a two-dimensional hypercube formed using the sign permutations ($\pm 1$) of the coordinates surrounding a sample point (open circle) at the origin. The closest radial interpolation center (filled circles) in each bin is selected. Bins are indexed using a $2^d$-bit integer.

## 2 Interpolation Method

The one-dimensional linear interpolation method is extended to arbitrary dimensions using bins to create a convex hull representation of the data around a sample point. The bins are formed using the sign permutations ($\pm 1$) of the coordinates, analogous to a $d-$dimensional hypercube. Within each bin the closest radial point from the set of normalized interpolation centers is chosen, and a convex hull is formed using the set of bins as the vertices. This is shown graphically in two dimensions in Fig. 1. An integer with $2^d$ bits contains the indices of each bin. The interpolation centers are normalized using the span of the parameter space. Data at the sample point is constructed by successively contracting the lowest dimension of the hypercube from $d \to 0$ using linear interpolation along the edges of the hypercube (*cf.* Fig. 2). This provides a direct monotone interpolation method with quadratic convergence.

Choosing the closest radial interpolation center within each hypercube bin removes difficulties due to data clustering. The current method is compared to the convex hull of a simplicial method for clustered data in Fig. 3. The hypercube binning naturally agglomerates the clustered centers and results in a regular polytope which does not require deconstruction to simplices. When an ap-

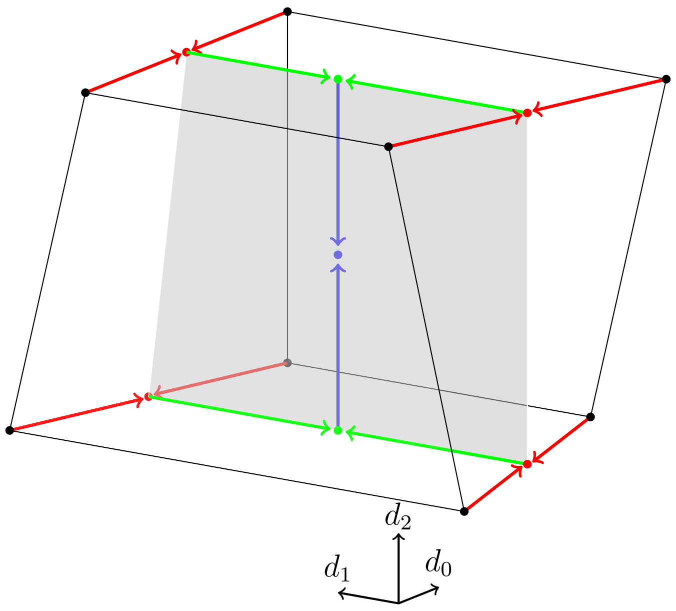


**Figure 2:** Contraction of nested hypercubes to the sample point using linear interpolation along each edge of the hypercube.

propriate interpolation center is not found within a bin the algorithm reverts to nearest-neighbor (zeroth-order) extrapolation for the affected dimensions, preserving monotonicity. Figure 4 demonstrates extrapolation in two dimensions for the case of pure extrapolation, and mixed extrapolation and interpolation. The important point is that extrapolation in this manner does not require special processing and the result remains monotone.

As the convex hull is constructed directly the algorithm has optimal memory usage, only requiring storage of the $dm$ data samples at each interpolation center. The algorithmic complexity is independent of the number of functionals being evaluated. Using a linear search to find the closest radial interpolation center in all hypercube bins the algorithmic complexity is $\mathcal{O}(2^d n)$. To determine the interpolation coefficients directly would require solving a linear system with $\mathcal{O}(2^{3d})$ complexity, however since we perform a contraction along each hypercube dimension individually the complexity reduces to $\mathcal{O}(2^d)$ to determine the interpolated values at the sample point.

It is straightforward to construct hash or tree search algorithms to reduce the complexity of the radial search, however these require care to handle general clustered data. As we search for the nearest radial point to our sample, we can bin the interpolation centers based on radial distance from an arbitrary point for the cost of an initial $\mathcal{O}(n \log n)$ pre-sorting of the data (*cf.* Fig. 5). The minimum distance from the sample point to the inner or outer radius of each bin is easily determined

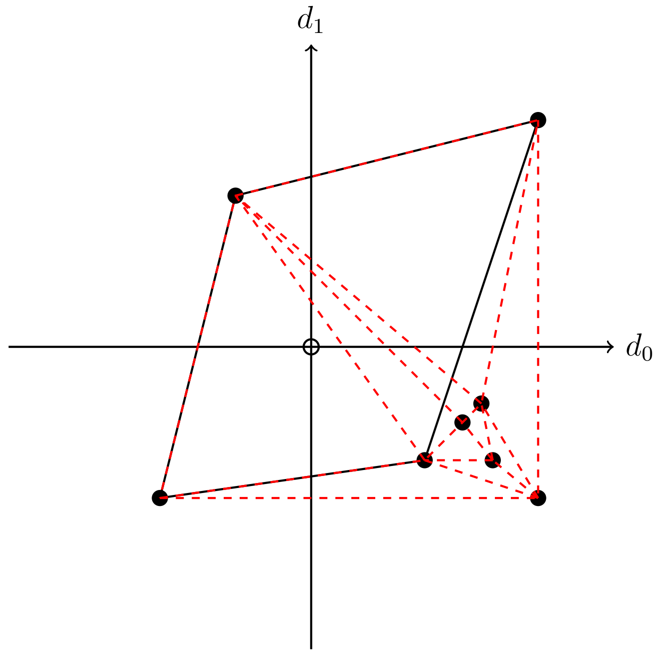


**Figure 3:** Convex hull due to agglomerating clustered interpolation centers to the nearest radial value (solid polygon), in contrast with a simplicial algorithm (dashed polygon).

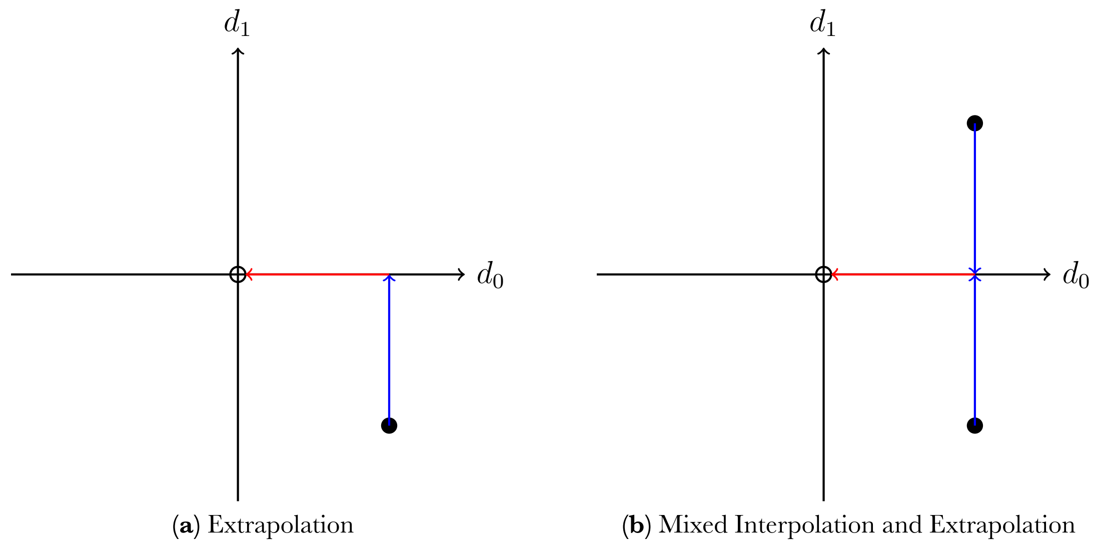


**Figure 4:** Extrapolation examples in two dimensions. When only a single donor is found (a), the value is extrapolated in $d_1$ and $d_0$. When sufficient candidates are found in $d_1$, but not $d_0$ (b), the value is first interpolated along $d_1$, then extrapolated along $d_0$.

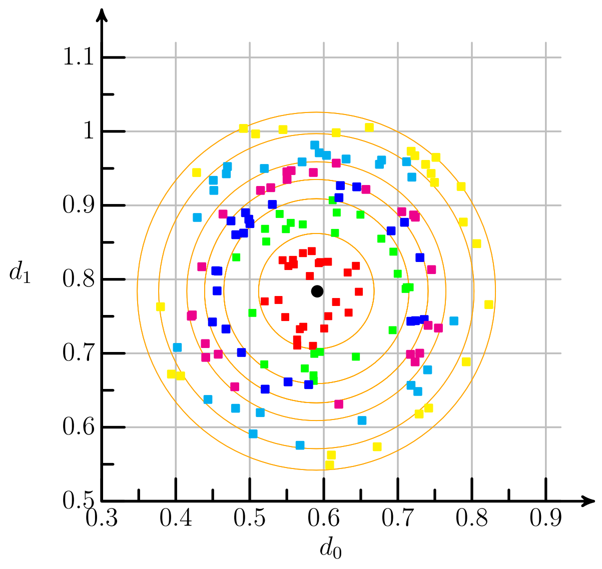


**Figure 5:** Normalized two-dimensional parameter space sorted into spherical bins based on distance from the approximate multivariate median (black circle). The colored squares are members of each bin, and the orange circles represent the extent of the spherical bins.

from

$$r_{min} = |\|\mathbf{d} - \mathbf{d}_\circ\| - r_{sphere}| \tag{1}$$

where $\mathbf{d}_\circ$ is the center of the spherical bins. As this point is arbitrary, we use an approximate multivariate median of the interpolation centers, which is straightforward to compute and performs well in practice. Using Eqn. 1 and the radial extent of the spherical bins we can provide a good initial guess and quickly ignore extraneous regions, resulting in a search algorithm with complexity $\mathcal{O}(2^d \log n)$.

As the algorithm outlined above requires minimal memory overhead, it is straightforward to construct a multithreaded reduction algorithm to determine the nearest radial centers in parallel. This reduces the algorithmic complexity further to $\mathcal{O}(2^d \log \frac{n}{p})$, where $p$ is the number of threads. The next section provides examples and timings for both a multicore CPU and graphics processing unit (GPU).

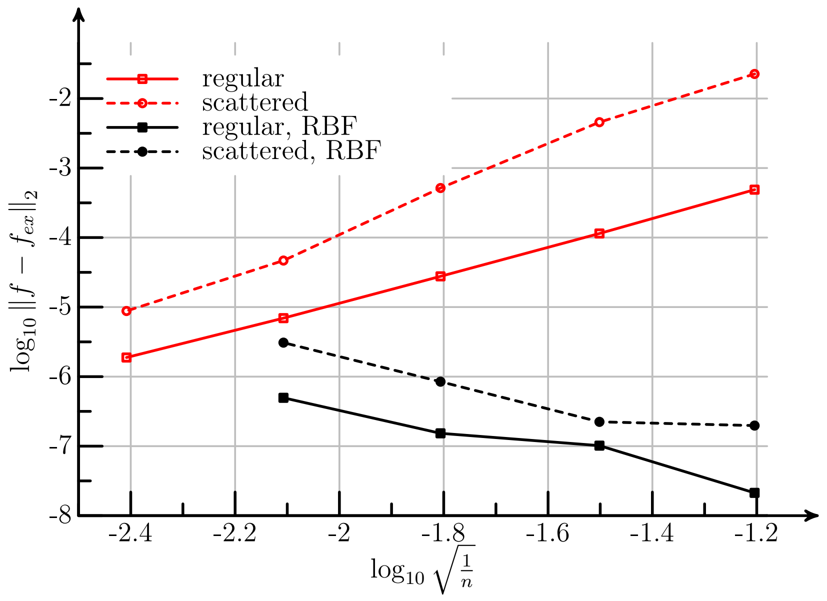


**Figure 6:** Convergence of the nested hypercube interpolation method for regular and scattered representations of a smooth functional. Comparison with a multi-quadric RBF method ($\phi(r) = \sqrt{r^2 + 1}$). The finest resolution is not included in the RBF results due to computational cost.

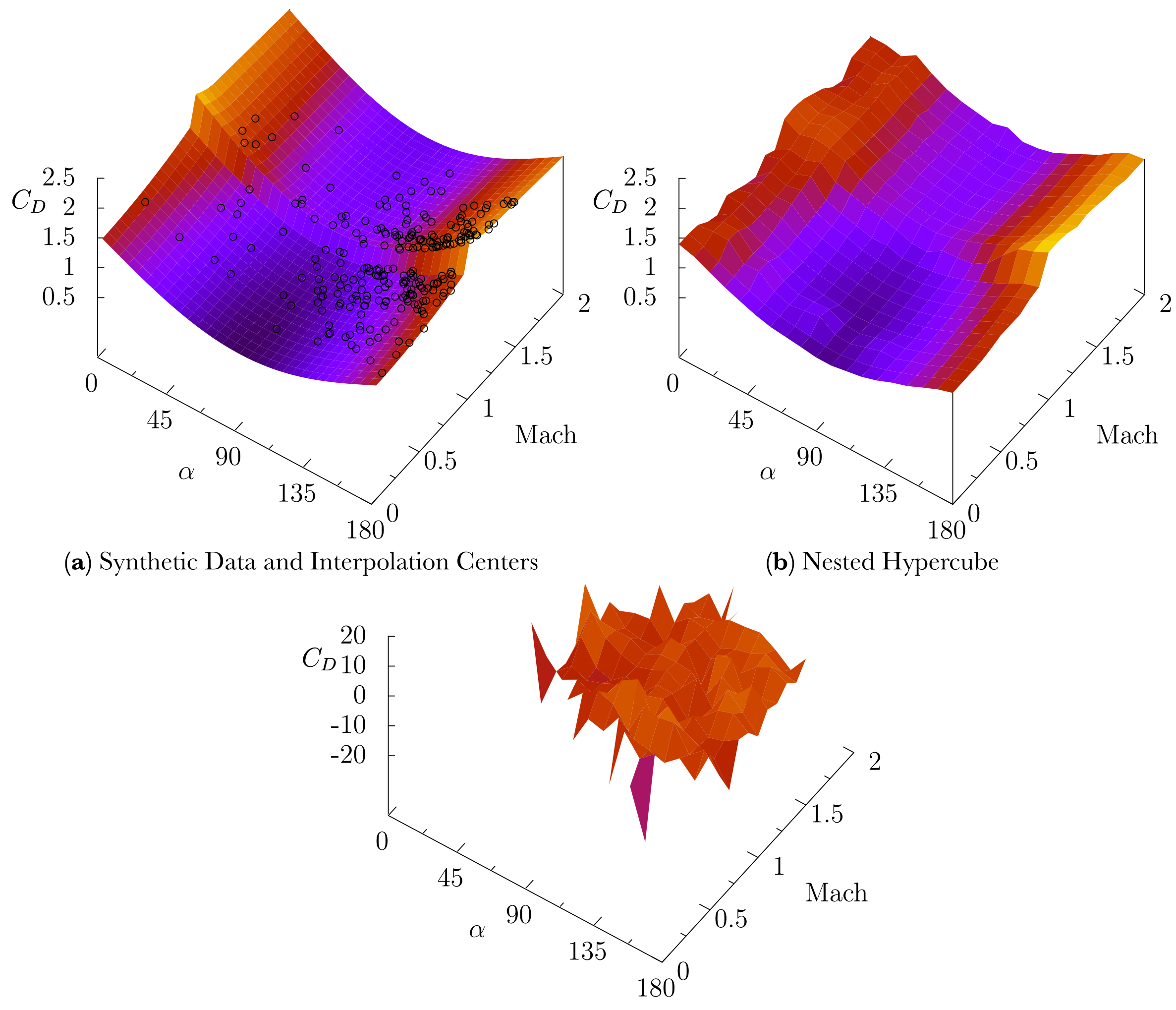
## 3 Numerical Examples

The formal accuracy of the interpolation method is demonstrated for smooth data using the two-dimensional test function,

$$f_{ex}(d_0, d_1) = a_0 + a_1 d_0 + a_2 d_0^2 + a_3 d_1 + a_4 d_1^2 + a_5 d_0 d_1 \qquad d_i \in [0, 1] \tag{2}$$

with the coefficients $a_i$ chosen from a random draw over $[-1, 1]$. One hundred random samples were taken, and the median value of the error over these samples was calculated from 100 trials. Convergence of the error with increasing density of the interpolation centers is presented in Fig. 6 for both a regular lattice, and scattered centers randomly drawn from a uniform distribution. Interpolation using a multi-quadric RBF is included for comparison. The current nested hypercube method demonstrates quadratic convergence for both the regular and scattered sets. As expected for this smooth functional, the global RBF interpolation method provides improved predictions over the local monotone scheme at coarse resolutions, with the increment decreasing with increasing resolution.

The interpolation method is tested using synthetic aerodynamic data based on a model for bluff-body drag variation with Mach number[8], and a sinusoidal variation of the loading with angle of



(**a**) Synthetic Data and Interpolation Centers (**b**) Nested Hypercube (**c**) Gaussian RBF

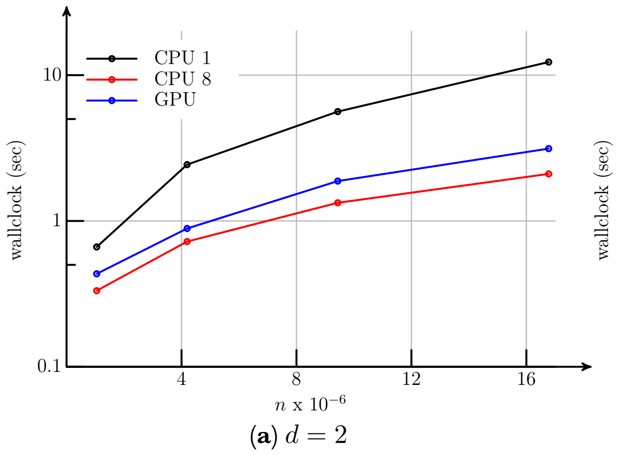
**Figure 7:** Reconstruction of a synthetic aerodynamic response surface from scattered interpolation centers using the nested hypercube and RBF ($\phi(r) = e^{-r^2}$) methods.

attack. The synthetic data is shown in Fig. 7a, along with 256 scattered interpolation centers taken from Gaussian sampling centered on the sonic line and 160° angle of attack. These scattered interpolation centers are used to reconstruct the response surface using the nested hypercube scheme in Fig. 7b, and a Gaussian RBF in Fig. 7c. The nested hypercube method provides a linear estimate of the response surface in regions with coverage from the interpolation centers, and a reasonable extrapolation outside the hull of the centers. The surface generated from a straightforward implementation of the Gaussian RBF is unsuitable for any application. Other choices for the RBF performed similarly.

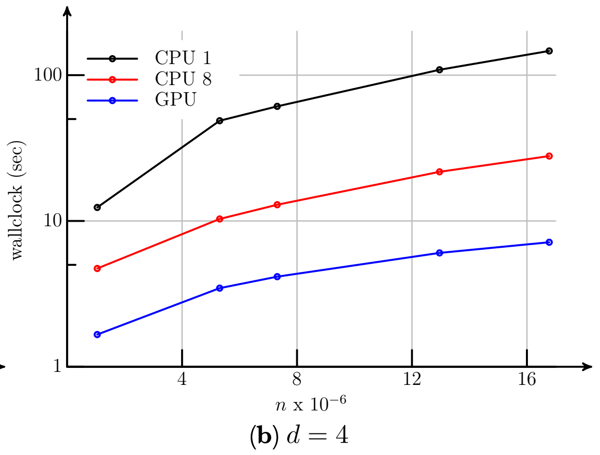
The computational performance and scalability of the algorithm is demonstrated in Fig. 8. An increasing number of interpolation centers are queried to interpolate 1000 samples for $d = 2$, 4, and 6. The results are presented for $m = 6$ linear functionals where the interpolation is exact to machine precision. The timings include both the radial search and actual interpolation. Results for both serial and parallel multicore search implementations are presented. Determining the closest radial interpolation center is implemented using standard multithreaded reduction methods. The interpolation centers within each spherical bin are divided into $p$ partitions of dimension $n_{bin}/p$, with the hypercube bins being treated serially within each partition. After finding the minimum in each partition in parallel, a serial global reduction across the partitions determines the final result. The implementation only performs the minimum reduction in parallel, so there is no expectation to approach ideal parallel scalability in this application. The GPU implementation is currently optimized for $d = 3$, and hence is penalized when $d < 3$. As expected, the results demonstrate that the hypercube implementation asymptotically does scale logarithmically with increasing number of interpolation centers for all dimensions, indicating that the radial search is the predominate cost. The empirical data indicates an overall scaling of roughly $\mathcal{O}(2^{d+1} \log n)$. We can interpolate six functionals through 17x$10^6$ centers for $d = 6$ in roughly 0.1 sec per sample using the GPU, and 0.15 sec per sample using a single Intel Nehalem 8-core node. For comparison, the parallel RBF implementation in [9] uses 106 wallclock seconds on 1024 cores to determine the weighting for 50x$10^6$ interpolation centers for a single functional in $d = 2$.

## 4 Summary

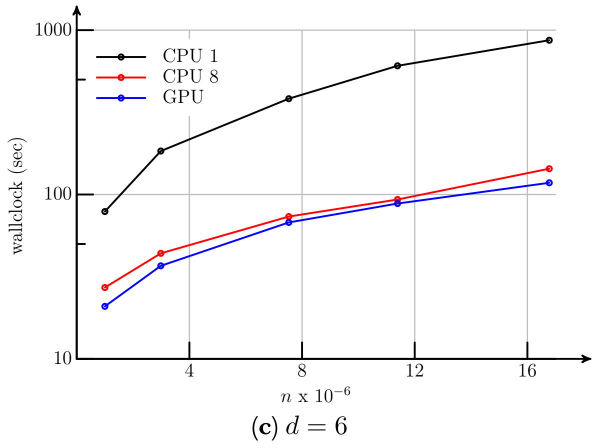
The nested hypercube interpolation method provides a robust and efficient method for querying clustered multivariate data with monotonicity constraints. The algorithm displays logarithmic scaling with increasing numbers of interpolation centers for practical problems. The primary application for the current work is the analysis of aerodynamic trajectory data. Generating aerodynamic performance data is relatively much more expensive than evaluating it, and the current method places the burden of cost where it can most easily be absorbed. The ability to rapidly and accurately interpolate scattered multivariate data provides the flexibility to utilize aerodynamic data generated by optimized or disparate sources without loss of fidelity. The method provides a general scheme for querying multivariate data, and also facilitates creation of structured response surfaces, for example in multi-fidelity uncertainty quantification.



(**a**) $d = 2$



(**b**) $d = 4$



(**c**) $d = 6$

**Figure 8:** Computational timings to interpolate 1000 samples for representative scaling studies. The CPU is an Intel Xeon W5580 3.2GHz, and the GPU is an NVIDIA Tesla C2070.

## References

[1] Alfeld, P., "Scattered Data Interpolation in Three or More Variables," in *Mathematical Methods in Computer Aided Geometric Design*, pp. 1–33, Academic Press, Boston, MA, 1989.

[2] Striepe, S.A., Way, D.W., Dwyer, A.M., and Balaram, J., "Mars Science Laboratory Simulations for Entry, Descent, and Landing," *Journal of Spacecraft and Rockets*, 43(2):311–323, 2006.

[3] Davis, J.L., Striepe, S.A., Maddock, R.W., Hines, G.D., Paschall, S., Cohanim, B.E., Fill, T.J., Johnson, M.C., Bishop, R.H., DeMars, K.J., Sostaric, R.R., and Johnson, A.E., "Advances in POST2 End-to-End Descent and Landing Simulation for the ALHAT Project," AIAA Paper 2008-6938, 2008.

[4] Fornberg, B. and Flyer, N., "Accuracy of Radial Basis Function Interpolation and Derivative Approximations on 1-D Infinite Grids," *Advances in Computational Mathematics*, 23:5–20, 2005.

[5] Fasshauer, G.E. and Zhang, J.G., "Preconditioning of Radial Basis Function Interpolation Systems via Accelerated Iterated Approximate Moving Least Square Approximation," *Computational Methods in Applied Sciences*, 11:57–75, 2009.

[6] Seidel, R., "Constructing Higher-Dimensional Convex Hulls at Logarithmic Cost per Face," in *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, 1986.

[7] Clarkson, K.L., Mehlhorn, K., and Seidel, R., "Four Results on Randomized Incremental Construction," *Computational Geometry*, 3(4):185–212, 1993.

[8] Murman, S.M., "Drag Behavior of Unconstrained Bluff Bodies," NASA Technical Memorandum TM-2010-216406, September 2010.

[9] Yokota, R., Barba, L.A., and Knepley, M.G., "PetRBF - A parallel O(N) algorithm for radial basis function interpolation with Gaussians," *Computer Methods in Applied Mechanics and Engineering*, 199:1793–1804, 2010.